Introduction to Programming with Python

Riding the Serpent

Anshul Nigham & Rob Tirrell

November 12, 2010

About Us

Anshul Nigham

- Works somewhere and does something.
- Look, it's relevant.
- And he's qualified! :)

Rob Tirrell

- Second-year grad student in Biomedical Informatics here at Stanford.
- Works in the Butte lab, an entirely "dry" lab (computers only – the only other equipment necessary is a coffee machine).
- First language was Python, spends most days writing code in R, Python, Ruby and C++.

What is Python?

- First released in 1991 by a Dutchman named Guido van Rossum (GvR).
- That's Self-Appointed Benevolent Dictator for Life (SABDFL) van Rossum to the rest of us.
- An interpreted, high-level language with flexible typing.
- Currently on its third major release... in other words, very mature.

A Satisfied User

> *"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language."*

– Peter Norvig, Director of Search Quality at Google and Computer Science Superstar

Other Satisfied Users

- **AstraZeneca** uses Python in drug discovery pipelines.
- **Phillips'** fabrication plants are managed in Python.
- **Industrial Light & Magic** (Star Wars) employs Python for process management.
- TODO: anecdotes from our work experiences.
- It may be new to you, but according to TIOBE's programming languages index, Python is the sixth-most popular in the world.

Good for Them, What's in it for You?

Power

- Python facilitates rapid development. It comes with a huge collection of software (libraries) for many purposes.
- There is a large, vibrant Python community/ecosystem.

Clarity

- Python is remarkably clear and readable compared to many other languages.
- It actually takes some effort to write difficult-to-understand programs.

Datatypes (1)

What are They?

- A **datatype** refers to a location in the computer's memory and the type of information stored there.
- Numbers can be of the integer datatype, like `4`, or a floating-point datatype, like `4.0`).
- Text uses the string datatype, like `'Four score and seven years ago...'`.
- True-false values are boolean datatypes, in Python these are `True` and `False`.

Datatypes (2)

More Advanced Datatypes

- Obviously, more complex programs require more complex datatypes (depending on the complexity, they may be referred to as 'data structures').
- The two most important in Python are lists and dictionaries.
- A list:

```
some_list = [12, 'monkeys']
```

- A dictionary:

```
some_dict = {
  'New York': 'A state in the US.',
  42: 'A completely irrelevant number.',
  'today_is_sunday': False
}
```

Datatypes (3)

Duck Typing

- The Python philosophy: if it looks like a duck, and it quacks like a duck, then it's a duck.
- In other words, variables are not restricted to contain a particular datatype. We can re-assign them to anything we want:

```
var = 44
var = 'The Life of Brian'
var = [
  'Episode IV: A New Hope',
  'Episode V: The Empire Strikes Back',
  'Episode VI: Return of the Jedi'
]
var = {
  'aardvark': 'a nocturnal burrower from Africa',
  'antbear': 'another name for an aardvark'
}
```

- Functions exist to coerce strings like `'1024'` into the corresponding floating-point or integer numbers (c.f. `int` and `float`), and so on.

Introduction to Functions and Methods (1)

Making Things Happen

- **Methods** and **functions**[1] are procedures that work on variables to transform or otherwise alter them.
- Change a string to all uppercase:

```
some_string = 'julius'
some_string = some_string.upper()
# some_string is now 'JULIUS'
```

- Access some entry (sometimes called an element) in a dictionary:

```
some_dict[42]
# >>> 'A completely irrelevant number'
```

[1] There is a distinction: methods are attached to and operate on their objects, while functions stand alone. Don't worry too much about this now, we'll come back to it later.

Introduction to Functions and Methods (2)

A Few More Examples

- Many of Python's datatypes support this sort of access (called indexing), which behind-the-scenes is just a regular method call:

```
some_string[1]
# >>> 'U'
```

- This is an important point: in Python, the first element of a collection is the "[0]th" one, available at `collection[0]`.
- Similarly, we can access the second element of a list with `some_list[1]`.

The Interpreter (1)

>>>

- As Python is an interpreted language (the computer reads, understands (compiles) and executes it on the fly, instead of reading and compiling ahead of time (as with C, C++, and many others).
- Because of this, we can use Python interactively, which is extremely helpful for designing and troubleshooting code.

IDLE

- IDLE is Python's interpreter shell, which we'll use to walk through examples throughout the course.
- For those of you with Macs, you can open a Terminal and launch Python by typing `python`.

The Interpreter (2)

The `print` Function

- Strings are surrounded by single (') or double (") quotes, which are equivalent
- `print` is a core Python function, which by default outputs text to the screen. It's somewhat special, in that parentheses are optional when using (calling) it.

A Longstanding Tradition

Two equivalent Hello Worlds:

```
print 'Hello World'
print('Hello World')
```

Brief Interlude: Setting Up KomodoEdit

KomodoEdit should already be installed on the provided computers, if not, visit http://www.activestate.com/komodo-edit/downloads to grab it.

Functions

Writing Your Own with `def`

- The `def` keyword is used to create new functions.
- Functions always `return` a value, which might be nothing (`None` in Python).

```
# This function has no parameters and...
def test_func_1():
  return 'returns a string.'
```

Functions (2)

```
# This function has no parameters and returns None.
def test_func_2():
  pass

# This function has one parameter - a number.
# It adds 10 and returns that value.
def add_10(parameter_1):
  return parameter_1 + 10
```

Write Your Own

Try writing a simple function that takes one parameter (a string) and uses `+` to display `'Hello <your string goes here>'`.

lgpay atinlay (1)

Pig Latin

Pig latin is a nonsense language, that can be generated easily for any English word, by moving the word's first character to the end and adding `'ay'`. Then, `pig_latinize('monster') = 'onstermay'`

String Indexing and Slicing

- Recall that we can access single elements of a collection using `[]`. We can also access longer parts of the string (which is a collection of characters) using **slices**.

```
green_eggs = 'green eggs and ham'
print green_eggs[6:10]
>>> 'eggs'
```

- Slices range from `0`, the first element, to `-1`, the last element.
- Omitting the start or end defaults to 'from the beginning' or 'to the end', respectively. Thus, `some_collection[:] == some_collection`.

Igpay atinlay (2)

Make Your Own Nonsense

- Using slicing, write a function that takes one parameter (a string) and returns the pig latinized version of the string.
- You'll have to make one slice, access one element of the string directly, and then add 'ay' at the end: `<all but the first character> + <the first character> + 'ay'`.

Igpay atinlay (3)

Improving Pig Latinization

You probably noticed that capitalized characters look out of place in the first version of the function. There are two string methods, `lower` and `capitalize`[2], that we can use to fix this problem.

- Modify your function so that it converts the pig latinized string to lowercase, then capitalizes it.
- You can do all of this in one line, but it is easier to read and understand if you store the intermediate pig latinized string in its own variables, like `pig_latinized_string`.
- Helpfully, you can 'chain' method calls, in other words, `some_string.lower().capitalize()` works!

[2] An exhaustive list of string methods is available at http://docs.python.org/library/string.html. All of the builtin libraries are well-documented, see http://docs.python.org

Dealing with Uncertainty (1)

Truth Testing

- Few programs are just a straight pipeline of unbiased transformations to data – most of the time, you'll want to branch out and make decisions based on input.
- Python has the `if`, `elif` and `else` keywords that provide this ability.

```
some_boolean = true
if some_boolean:
  print 'Truly today is a special day!'
else:
  print 'Back to the grind, peasant.'
```

Dealing with Uncertainty (2)

Equality

- You can also use ==, <, <=, and so on.
- The instructions following an `elif` keyword are only triggered if no previous conditions are met and the `elif` condition is.

```
strength = 15
if strength >= 15:
  print 'You pull the sword from the stone.'
  print 'All hail King Arthur!'
elif strength >= 10:
  print 'Solid effort.'
  print 'But you were born to work in the mines.'
else:
  print 'Nope! Not even close.'
```

Looks like it's deep, dark and dirty for this guy.

Case Study: The Candy Shoppe (1)

A Profitable Business Proposition

- You run a candy shoppe, selling all variety of popsticks, licorice screamers, fireballs and the like.
- Little kids are big business you, and these little sugar fiends are spending their parents money and are not very sensitive to changes in price. Being a businessperson first and foremost concerned with the bottom line, you want to charge anyone less than `14` years old a little bit extra (`25%` over regular price).
- But you're not a monster, and if the customer's blood sugar is below `2.5`, you'll give him or her a `15%` discount (with no additional penalty for being less than `14`).
- You want this to be done transparently, so you will add a function to your cash machine that jacks up the price depending on the customer's information (which is stored in the customer's loyalty card).

Case Study: The Candy Shoppe (2)

Customer Information

- Each customer's loyalty card has several attributes encoded in the magstripe.
- These attributes are loaded into the cash machine when you scan the card, and are available to the software as a dictionary.

```
customer_info = {
  'name': 'Rob Tirrell',
  'email': 'rpt@stanford.edu',
  'age': 24,
  'blood_sugar': 5.0, # mmol / L
  'likes_dark_chocolate': False,
  'astrological_sign': 'Gemini'
}
```

Case Study: The Candy Shoppe (3)

Implementation

- Recall that we access dictionary values by their key, so for the above example, `customer_info['age']` will return 24.
- As we might except, we can make comparisons with dictionary values directly, without storing the value in a separate variable:

```
if customer_info['age'] > 65:
    print 'Senior citizen discount!'
```

- We want to write a function, that given the purchase price and a customer's information, 'adjusts' the price appropriately. It should look something like:

```
def compute_price(base_price, customer_info):
    pass # Fill in here!
```

Importing and Using Modules (1)

- So far, we've only played with the core of Python, so let's dive into the standard library (the suite of useful software that comes with Python).
- To load any one of these libraries, we use `import` statements:

```
import random
# Generate an integer from 1 to 6 (inclusive).
random.randint(1, 6)
# Generate an integer from 1 to *5* (inclusive).
random.randrange(1, 6)
```

Importing and Using Modules (2)

Who's Feeling Lucky?!

- Let's use random to pick the winner of a lottery. First, define the players:

```
lottery_players = [
  'Rufus',
  'Sun',
  'Tania',
  'Boris',
  'Rocky',
  'Daisy'
]
```

- Then, picking a winner with `random` is stupid simple:

```
print random.choice(lottery_players)
```

Importing and Using Modules (3)

Shufflin'

- How about one more quick example with the same people as before. We want to split the people into two teams, so we can sell tickets to a 3v3 wrestling extravaganza to pay off their gambling debts.
- Again, ask and `random` provides the `shuffle` function, which randomizes the ordering of a list **in-place** (that is to say, it returns `None`).

Importing and Using Modules (4)

Picking Sides

- Let's shuffle the list, and then save the number of players in a variable `number_of_players` using the builtin `len` function.[3].

```
number_of_players = len(lottery_players)
print 'Team 1:'
print lottery_players[:number_of_players / 2]
print 'Team 2:'
print lottery_players[number_of_players / 2:]
```

[3] `len(object)` is really just a proxy for the method call `object.__len__()`. This is also the case for operators, e.g., `object[i]`, which just calls `object.__getitem__(i)` behind the scenes, and `object + 1` becomes `object.__add__(1)`.

Choose Your Own Adventure

We have a couple of choices for what to do next. We can:

- Write a program that analyzes a document for duplicated words, featuring a brief introduction to natural-language processing (NLP) theory and techniques.
- Write a program that uses Google web services to 'fight' terms against each other (in the spirit of Google Fight).

So We've Sold You on Python, What's Next? (1)

- There are **tons** of freely-available resources on the internet to continuing experimenting and learning. The best place to start is probably Dive Into Python 3 (http://diveintopython3.org) by Mark Pilgrim. As the name indicates, it covers Python 3 (we are using 2.7, some things differ but most of what we've gone over is the same).
- We haven't even nearly covered the standard library (the stuff that comes with every version of Python). Honestly, I haven't used or even heard of most of it.
- Python has builtin modules for scraping from the internet (`HTMLParser`), (extremely advanced) string searching and manipulation (the strange world of regular expressions, in the `re` module), creating, deleting and managing files and directories (`os` and `shutil`), creating and extracting compressed files (`gzip`, `bz2`, `zipfile`, `tarfile`), interacting with FTP servers (`ftplib`), and on and on and on $\rightarrow \infty$.

So We've Sold You on Python, What's Next? (2)

Stepping away from builtin functionality, there are many other mature libraries in the wild:

- If you want to write websites, check out Google's App Engine (http://code.google.com/appengine/). It provides free hosting services and holds your hand along the way (that's a good thing!). For more advanced websites, look into the Django Project (http://www.djangoproject.com/) (App Engine is actually based on Django).
- There are libraries for image manipulation (the Python Imaging Library, http://www.pythonware.com/products/pil/), numerical and scientific calculation (NumPy/SciPy, http://numpy.scipy.org/), bioinformatics (BioPython, http://biopython.org/wiki/Biopython), plus a whole slew of others for most every purpose imaginable.